REMARKS

Claims 1-53 stand rejected under 35 U.S.C. § 112, second paragraph, as being indefinite. Particularly, the Office Action submits that the language in independent claims 1, 27, and 35, "wherein each of the plurality of intervals of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time wherein the amount of time is defined by at least one of a predetermined time interval, predefined number of instructions that are to be run, and a time having a length based on a predetermined metric", is indefinite. Applicant respectfully traverses the rejection, because the rejection construes the claim language based on an incorrect reading of the present specification, and it fails to consider the explicit language in the claims.

The Office Action begins its reasons for rejection at page 11, lines 7-9 of the present specification, which states, "For example, an interval may be thought as a slice of time…an interval may be…a time interval, an instruction interval, and/or a metric-based interval." Based allegedly on this language, the Examiner concludes that he "does not consider a 'predefined number of instructions to be run' as the same as 'an instruction interval'." Then, the Office Action attempts to define "an instruction interval" and alleges that it may have more than one meaning.

However, again, the claim language at issue is "a defined amount of time wherein the amount of time is defined by...predefined number of instructions that are run", not "an instruction interval". The Office Action's analysis allegedly considers the

claim language, but inappropriately chooses only alternate language to interpret the claim.

Further, the Office Action states, "It appears the number of instructions is collected after the execution of the interval instead of being predefined." This ignores the explicit claim language. The claims require that the instructions are run over a "defined amount of time wherein the amount of time is defined by at least one of a predefined time interval, predefined number of instructions that are to be run, and a time having a length based on a predefined metric" (emphasis added). There is no reasonable reading of the phrase "predefined number of instructions that are run" in which the number of instructions that are run are not predefined. Yet, the Office Action offers such a reading. There is no explanation given for simply removing the word "predefined" from the claim.

Further, in interpreting the claim phrase "predefined number of instructions that are to be run", the Office Action relies on a section of the application that refers to tracking statistics, not defining time. The section provided in page 11, lines 3-12 describes what an interval is, and how time may be defined. One of ordinary skill in the art having reference to the present specification would appreciate that the predefined number of instructions refers to how time is defined, and that the time can be defined by a predefined number of instructions or by a time having a length based on a predefined metric, among other ways.

However, the Office Action instead relies on page 11, line 22 – page 12, line 3, which states, "Exemplary statistics include frequency, "number of instructions executed, number of memory accessed…". The description of statistics discussed in this

section is separate from the definitions of time that should be used to construe the claim language that defines time.

Apparently, the Office Action reaches this point by citing the phrase "instruction interval" in the specification and then concluding that "the examiner does not consider 'a predefined number of instructions that are to be run' as the same as 'an instruction interval'". Applicant submits that 'instruction interval' as used in the present specification, in the context of being an example of a slice of time (which is how it is used in the cited portion of page 11, lines 3-12), refers to a defined number of instructions, as that is how time would be measured using instructions.

It is important to understand the difference between instructions used for defining an amount of time and instructions used for tracking a statistic for a component. For example, as provided in page 11, line 13 – page 12, line 3, the "number of instructions executed" is an example statistic (a hardware independent metric). This statistic, an output, is tracked for a program component, as provided in the preceding paragraph in page 11. An example program component is an instruction.

Thus, for a particular instruction, an example statistic that is tracked may be the number of those instructions that are run during a period of time, and this time may be defined as a predefined number of instructions that are run. As a nonlimiting example, suppose one wanted to track how many times a particular instruction at line 100 is run during a time defined by intervals of 500 million instructions. The instruction at line 100 would be a component (more particularly, a program component), the time defined by a predefined number of instructions (500 million) would be a defined slice of time, and the

number of times the instruction at line 100 executed during that time would be a statistic for the component.

Note that in this example, a predefined number of instructions is used to define a time, as described in page 11, lines 3-12 of the specification, and a number of instructions is (separately) used as a statistic to track a component (a particular instruction – line 100), as described in page 11, line 12 – page 12, line 3 of the specification. The predefined number of instructions used to define a time is an input, and the number of instructions for line 100 is an output.

Thus, the claimed definition of time is consistent with the section of the specification cited by Applicant, page 11, lines 3-12, and clearly and explicitly requires that the time be defined and that the number of instructions be predefined. However, instead of interpreting the explicit language of the claim, the Office Action on its own removes the claim language "predefined number of instructions that are to be run" from the claim interpretation and substitutes it with "an instruction interval". The Office Action then cites a different portion of the specification that refers to a number of instructions executed as an example of a statistic, and uses this to interpret the specification language "an instruction interval" as an interval from a first point in a program to a second point in a program. Finally, the Office Action declares the claim language to be indefinite, because it can somehow be interpreted differently from the actual claim language, though this interpretation fails to refer to the claim language itself.

Applicant included the explicit phrase "predefined number of instructions that are to be run" in the independent claims to clarify the claims, especially with respect

to cited reference Goodnow. Applicant also submits in the amendment herewith new claims 54-57, which again attempt to define a slice of time. However, such amendments are pointless if the Office Action can simply rewrite the claim language, and then declare the new language indefinite.

The Office Action makes a similar misinterpretation with respect to another claimed definition of time, time having a length based on a predefined metric. Particularly, the Office Action states:

> … the examiner does not consider "a time having a length based on a predefined metric" as the same as 'a metric-based interval' as discloses [sic] in the specification. A predefined metric can be interpreted as any measurement unit, including, for example a program block, a function in a program, or a code segment, while a metric-based interval is interpreted as an interval measured using metric-based system, such as seconds. Therefore "a time having a length based on a predefined metric" can be interpreted as the time based on executing a program block, a function in a program or a code segment.

The claim language at issue is "a defined amount of time, wherein the amount of time is defined by…a time having a length based on a predefined metric". Again, the Office Action fails to consider all of the words used in the claim or the pertinent part of the specification, and thus selectively interprets the claim language by substituting other language from the specification.

The actual claim language finds clear support in the specification. In page 11, lines 7-9, for example, it is stated that an interval may be thought of as a slice of time, though an interval may be, for example, a time interval, an instruction interval, and/or a metric-based interval. One of ordinary skill in the art having reference to the present

specification would appreciate the meaning of the word "metric". Thus, the metric-based interval used in page 11, lines 7-9, as cited by Applicant, is an example of such an interval; that is, a slice of time, having its length depending on a metric. This is consistent with the claim language "a time having a length based on a predefined metric".

By comparison, a metric, as discussed separately on page 11, lines 21-22, can also be used in the context of a statistic tracked for a component, which may be a hardware-based metric or a hardware-independent metric. This metric is used to define an output. It thus would be understood by those of ordinary skill in the art having reference to the present specification that a metric can be used to set a period of time, and a metric can also be used to provide a statistic for a component.

Given the support in the specification, the claim language requires "a time having a length based on a predefined metric". Note the explicit words used: "a time", "a length", "predefined". It is not reasonable to interpret this claim language as defining anything other than a time (an input). The clear language of the claim is that a time is defined as a time having a length based on a predefined metric; that is, a predefined metric is used to determine when a sufficient time has passed.

However, the Office Action ignores this language, relying instead on the language "metric-based interval", and proceeds to interpret this without any context as simply "an interval measured using metric-based system, such as seconds". The Office Action then concludes, "It is for this reason that one interpretation of the claim language can be different from what is being presented in the specification." One problem with this argument is that the Office Action supports it by citing an interpretation of language

that is not in the claim. It is much easier, but improper, to find that claim language can be interpreted more than one way when one is actually interpreting different language each time, especially in a vacuum.

Accordingly, Applicant respectfully submits that claims 1-53 (and new claims 54-57) are definite under 35 U.S.C. § 112, second paragraph. Reconsideration and withdrawal of the rejection is respectfully requested.

Claims 1-8, 12-17, 26, 43-48, 50, and 52 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Goodnow. Applicant respectfully traverses the rejection.

Given a proper interpretation of the claims, Goodnow fails to teach or suggest at least identifying a behavior of a computer program over each of a plurality of intervals of execution based on a tracked statistic for a program component, or comparing at least one identified behavior for at least one interval of execution to another interval of execution to determine similarity between the intervals of execution, where each of the plurality of intervals of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time, and wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric.

As repeatedly submitted by Applicant, Goodnow identifies behavior and compares identifies behavior of code segments, which are always defined by specific lines of code, not by time. Even though in Goodnow code segments may be identified as two functions within either the same program or different programs or any other defined designations such as, but not limited to blocks, Lvalues or statements, these code

segments are always defined by, and restricted to, particular lines of code. Code segments are defined by specific lines of code (C4, L42-45; C4, L62-65). Functions are defined by specific lines of code (Table 1; C5, L9-15; C10, L17 ("A function corresponds to a particular collection of blocks")). Blocks are defined by specific lines of code (C9, L13-14). Lvalues identify specific lines of code (an assigned location). Statements identify specific lines of code (by definition). Even when Goodnow relies on a code extractor to obtain code segments, as described in the related patent 5,574,837 incorporated therein by reference, these code segments are extracted (generated) and sent as specific lines of code. Nothing in column 4, or anywhere else in Goodnow (including the patent incorporated therein by reference), contradicts this.

To define intervals of execution by time in Goodnow would undermine the very nature of Goodnow's invention, because it would not allow evaluation and comparison of specific, identifiable pieces of code. The purpose of Goodnow's invention is to test particular lines of code so that they can be incorporated into one or more other programs. If the code segments are not defined by particular lines of code, there is nothing in particular to test. All of the testing described in Goodnow is performed with respect to code segments, not intervals of execution defined by time.

The Office Action alleges that the claim phrase "a time having a length based on a predefined metric" can be interpreted as the time based on executing a program block, a function in a program or a code segment. Using this interpretation, the Office Action finds that Goodnow's teaching of running a code segment reads on the claims. However, even if the time based on executing a program block, a function, or a

code segment is used as the defined time, Goodnow still fails to teach or suggest all claim features.

Note that claim 1 requires "running code of the computer program over a plurality of intervals of execution, wherein each of the plurality of intervals of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time, wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric". Behavior among this plurality of intervals is then compared.

Thus, assume for the sake of argument that the time for each interval is defined based on executing a code segment (i.e., the code segment is used as a metric). Goodnow teaches executing the program, but does not teach or suggest running the code over a plurality of intervals of execution, wherein each of the plurality of intervals of execution is defined by continuous instructions of the code in program execution over an amount of time, wherein the amount of time is defined based on executing a code segment.

If a particular code segment is used as a metric, as submitted by the Examiner, according to claim 1, the time of each interval is defined as the length to run that particular code segment. Goodnow does not teach or suggest running a program over a plurality of intervals of execution, where each interval is defined as the period of time needed to run that specific code segment, and then comparing the behavior of these

intervals of execution as required by the claim. Goodnow, instead, runs its code segment once and evaluates the results (dynamic assessment).

Accordingly, if the time to run that complete code segment from beginning to end is used as a metric, there would be exactly one interval, because when the code segment has been run to completion, the simulation stops. The code segment can be compared to other code segments, for example if the entire program was run to completion, but that would require a different metric for each code segment, and thus running one code segment and then running another would not be running the program over a plurality of intervals of execution, where each interval is defined as the period of time needed to run a specific code segment (the assumed metric).

Suppose, on the other hand, the time is defined as the time needed to run the code segment alone (e.g., the code segment had x number of instructions, so an interval is defined as every x instructions), and more than just the code segment was run, such as if the complete program is run. Goodnow still fails to teach running a plurality of intervals defined by the time to run that particular code segment and comparing behavior among these intervals as claimed (in other words, Goodnow does not teach comparing behavior among intervals of every x instructions). The behavior considered in Goodnow (during dynamic assessment) is limited to the behavior resulting from running each of the code segments individually. This is why, as Applicant has previously submitted, Goodnow's teaching is limited to comparing code segments, not intervals.

Thus, Goodnow's invention still does not teach or suggest identifying behavior over each of a plurality of intervals of execution of arbitrary sections of

execution defined by time, or comparing at least one identified behavior for at least one interval of execution to another interval of execution to determine similarity between the intervals of execution, as defined in independent claim 1 and dependent claims 2-8, 12-17, 26, 43-48, 50, and 52. Applicants thus respectfully request reconsideration and withdrawal of the rejection.

Claims 9-11, 22-25, 27-33, 35, and 36 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Goodnow in view of a publication by present inventor Calder, "Time Varying Behavior of Programs" ("Calder"). Applicants respectfully traverse the rejection.

Regarding dependent claims 9-11 and 22-25, Applicants respectfully traverse the rejection for at least the reason that Goodnow fails to teach or suggest at least identifying a behavior of a computer program over each of a plurality of intervals of execution based on a tracked statistic for a program component, or comparing at least one identified behavior for at least one interval of execution to another interval of execution to determine similarity between the intervals of execution, where each of the plurality of intervals of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time, and wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric, as defined in independent claim 1. Applicants' arguments regarding independent claim 1 and Goodnow are incorporated herein.

Regarding independent claim 27 and dependent claims 28-33 and 35, Applicant respectfully traverses the rejection for at least the reason that Goodnow fails to teach or suggest at least identifying a behavior of a hardware-independent metric within at least one arbitrary section of execution, and classifying each of the at least one arbitrary section of execution according to the identified behavior into clusters of behavior, where each of the arbitrary sections of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time, and wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric. For similar reasons as stated above regarding claim 1 and Goodnow, the reference fails to teach or suggest at least identifying behavior of a hardware-independent metric within at least one arbitrary section of execution.

Further, any clustering performed by Goodnow is with respect to functions that, again, are defined by specific lines of code, not by time (for example, see C13, L18, describing a cluster interface for functions f_0, f_1, f_2, f_3, and f_4, all defined by particular lines of code), even as the time is defined in the hypothetical presented in the Office Action. Put another way, Goodnow does not consider intervals having a defined time in its analysis, regardless of how time is defined, it only considers the running of particular lines of code (from the beginning of a code segment to the end). There are no comparisons of behavior, and no clustering that takes place among such intervals. All clustering takes place only among code segments.

With respect to independent claim 35 and dependent claim 36, Applicants respectfully traverse the rejection for at least the reason that Goodnow fails to teach or suggest at least identifying a behavior of a hardware-independent metric for each of a plurality of intervals of execution, or comparing the identified behavior of each of the plurality of intervals to identified target behavior over full execution of at least a portion of a computer program, where each of the plurality of intervals of execution is defined by continuous instructions of the code in program execution order run over a defined amount of time, and wherein the amount of time is defined by at least one of a predefined time interval, a predefined number of instructions that are to be run, and a time having a length based on a predefined metric. Goodnow fails to teach or suggest identifying a behavior of a hardware-independent metric for each of a plurality of intervals of execution for similar reasons as that stated above regarding claim 1.

Additionally, Goodnow fails to teach or suggest comparing identified behavior of each of the plurality of intervals to identified target behavior over full execution of at least a portion of a computer program. In addition to the reasons stated above with respect to time versus code segments, there is not teaching in any event in Goodnow of comparing behavior of even a single code segment to behavior over full execution of at least a portion of a computer program. Code segments in Goodnow are always compared to other code segments.

Further, with respect to claims 9-11, 22-25, 27-33, 35, and 36, secondary reference Calder fails to remedy the deficiencies of Goodnow regarding the claims. Calder is cited for simulating execution over intervals of 10 million or 100 million

instructions. However, Calder does not teach or suggest tracking a statistic for a program component for intervals of execution, as defined in independent claim 1, identifying behavior of a hardware-independent metric for arbitrary sections of execution, as defined in independent claim 27, or identifying behavior of a hardware-independent metric for each of a plurality of intervals of execution, as defined in independent claim 35. Instead, Calder specifically teaches identifying behavior based on hardware benchmarks (for example, see page 4, describing benchmark data such as percent Register Update Unit (RUU) occupancy, cache miss rate, branch prediction miss rate, address prediction miss rate, and value prediction miss rate).

By contrast, embodiments of the present invention allow behavior intervals of execution or arbitrary sections of execution (as defined in the claims) to be identified and compared, either to one another or to an overall behavior of a program, based on a hardware-independent metric, such as the code itself (e.g., a program component). More particular embodiments of the present invention (for example, as defined in one or more dependent claims) provide that execution of a computer program can be classified into phases based on such behavior, or compared to the behavior of a program (or a portion thereof) as a whole.

Note, however, that a program component, such as but not limited to a basic block, is an example of a metric to evaluate behavior of an interval of execution or arbitrary section of execution in embodiments of the present invention, but this should not be confused (as it is in the Office Action's 35 U.S.C. §112, second paragraph analysis) with how the interval of execution or arbitrary section of execution itself is

defined. By contrast, Goodnow considers only a specific block of instructions for execution and comparison, and Calder only uses a hardware metric to evaluate intervals of instructions.

Note also that reference Calder does show examples of running a program over intervals of execution, and this reference illustrates why such intervals are completely different from the individual code segments used in Goodnow. In the same manner that the Office Action refers to Calder to allege what is known in the art, Applicant submits that one of ordinary skill in the art having knowledge of both Goodnow and Calder would also appreciate the difference between the intervals used in Calder and the code segments used in Goodnow. This also illustrates why one of ordinary skill in the art would not look to Calder to modify Goodnow, or vice versa.

For at least these reasons, Applicants respectfully submit that claims 9-11, 22-25, 27-33, 35, and 36 are allowable over the references of record, including Goodnow and Calder. Applicants thus request reconsideration and withdrawal of the rejection.

Claim 34 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over Goodnow in view of Calder, and further in view of Clark. Claims 18-21, 41, 49, 51, and 53 stand rejected over Goodnow in view of Calder, Clark, and Baker. Applicants respectfully traverse the rejections for at least the reasons stated above regarding independent claims 1, 27, and 35, and for at least the additional reason that references Clark and Baker do not remedy the deficiencies of Goodnow and Calder regarding the claims. Applicants thus request reconsideration and withdrawal of the rejection.

For at least the above reasons, Applicants respectfully submit that this application is in condition for allowance, which is respectfully requested. The Examiner is invited to contact Applicants' attorney if an interview would expedite prosecution.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By: /Arik B. Ranson/
Arik B. Ranson
Registration No. 43,874

Customer No. 24978
April 23, 2009
300 South Wacker Drive
Suite 2500
Chicago, Illinois 60606
Telephone: (312) 360-0080
Facsimile: (312) 360-9315